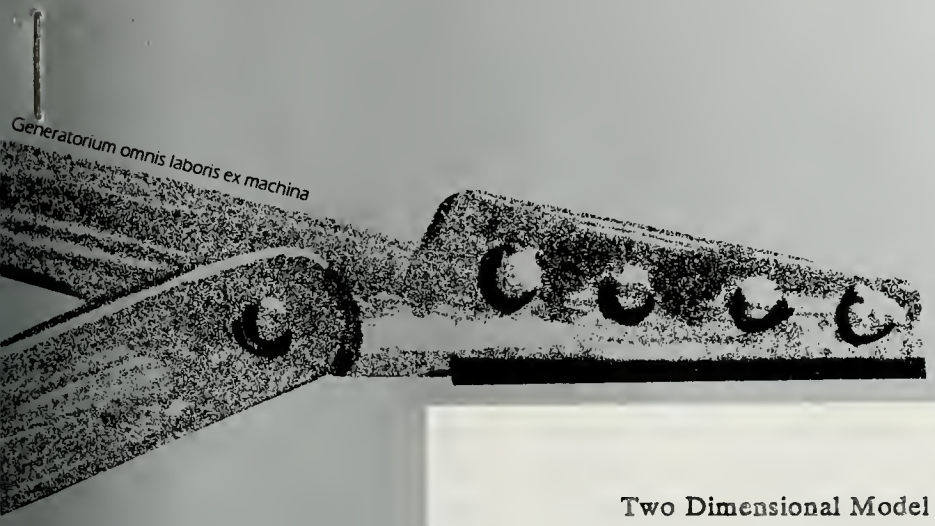
Robotics Research Technical Report



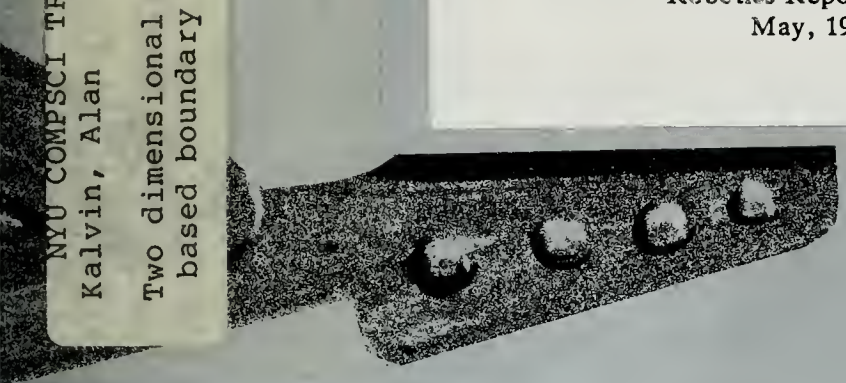
Two Dimensional Model Based
Boundary Matching Using Footprints

by

Alan Kalvin, Edith Schonberg,
Jacob T. Schwartz, and Micha Sharir

Technical Report No. 162
Robotics Report No. 44
May, 1985

NYU COMPSCI TR-162
Kalvin, Alan c.1
Two dimensional model
based boundary



New York University
Courant Institute of Mathematical Sciences

Computer Science Division
251 Mercer Street New York, N.Y. 10012

COURANT INSTI... LIBRARY
251 Mercer St. New York, N.Y. 10012

Two Dimensional Model Based Boundary Matching Using Footprints

by

Alan Kalvin, Edith Schonberg,
Jacob T. Schwartz, and Micha Sharir

Technical Report No. 162
Robotics Report No. 44
May, 1985

New York University
Dept. of Computer Science
Courant Institute of Mathematical Sciences
251 Mercer Street
New York, New York 10012

Work on this paper has been supported by Office of Naval Research Grant N00014-82-K-0381, National Science Foundation Grant No. NSF-DCR-83-20085, and by grants from the Digital Equipment Corporation, the Sloan Foundation, the System Development Foundation, the IBM Corporation, and the US-Israeli Binational Science Foundation.

Two Dimensional Model Based Boundary Matching Using Footprints

Alan Kalvin, Edith Schonberg, Jacob T. Schwartz, and Micha Sharir[1]

Robotics Activity
Courant Institute
New York University

ABSTRACT

A technique for geometrically hashing two-dimensional model objects is described. Used in conjunction with other methods for recognizing partially obscured and overlapping objects, this technique, which is based on use of an artificially generated attribute of an object called its *footprint*, enables us to recognize overlapping 2-dimensional objects selected from large databases of model objects without significant performance degradation. Experimental results from databases of size 48 and 100 are presented.

1. Introduction

The goal of *model-based* object recognition is to identify a given object as one of a collection of known model objects. In complicated versions of this problem, the object to be recognized may be partially occluded, or several objects to be identified may overlap. Techniques for solving these object recognition problems in the 2-dimensional case are presented in [9]. The recognition algorithm described there works by matching, i.e. identifies an object by matching all model objects against the boundary curve of the given object and choosing the best match. Therefore as the database of model objects grows large, the performance of that algorithm degrades linearly.

This report describes a technique that can be used to radically reduce the number of models used by the matching algorithm; that is, given a large list of models and an object O to be identified, we are able to obtain a small set of *candidate models* that potentially can match O. This technique, which is based on geometric hashing, is sublinear in the number of models so that we are able to work effectively with a large database of models. The object attribute that we form for purposes of hashing will be called a *footprint*. Experimentation with databases of sizes 48 and 100 models has shown that the footprint technique is good enough to identify observed partially occluded

[1]Work on this paper has been supported in part by Office of Naval Research Grant N00014-82-K-0381, and by grants from the Digital Equipment Corporation, the Sloan Foundation, the System Development Foundation, the IBM Corporation, and by National Science Foundation CER Grant No. DCR-8320085. Work by the fourth author has also been supported by a grant from the U.S.-Israel Binational Science Foundation.

or overlapping objects uniquely in most cases. Once a small set of candidate models has been selected using footprints, the role of the matching algorithm described in [9] becomes to confirm the identification.

Our recognition experiments begin with digitized raster images that are slightly corrupted by noise. Raster images of objects that are to be used as models are processed and appropriate information about these objects is stored in a model database. A raster image of a composite scene consisting of one or more overlapping model objects (we shall also call such a scene a *puzzle*) is subsequently processed in identical fashion in preparing to identify its component pieces. In addition to identifying the component objects that make up a puzzle, the matching algorithm determines the position and orientation of each object. An example of an overlapping object puzzle constructed from a database of model objects is shown in Figure 1.1; the model database from which this puzzle is constructed is shown in Figure 1.2.

Other boundary matching techniques include the method presented in [8], which finds the position and orientation of objects using a compact representation of the boundary of an object called its *concurve*; the method described in [1], which uses a polygonal boundary representation of objects to match straight line segments; and the method described in [10], which matches pairs of template segments against pairs of image segments using a slope angle-arclength graph representation of objects and probabilities. A different approach to the overlapping object identification problem is based on comparisons of local object features such as holes and corners ([2], [4], [5], and [7]). Another identification method presented in [3] is based on stochastic labeling.

The rest of this report is organized as follows. Section 2 reviews the object recognition, or *matching*, algorithm presented in [9]. This matching algorithm assumes that all boundaries have been smoothed and are represented as sequences of evenly spaced points. The pre-processing of the digitized images necessary to obtain objects in this form is also described. Our footprint hashing technique for pruning the number of candidate models is detailed in Section 3. Section 4 presents results obtained in experiments with model databases of sizes 9, 48, and 100. Various refinements that were needed to obtain good results are described. Possible enhancements of our present techniques are described in Section 5.

2. Matching Objects

In this section we review the object recognition algorithm described in [9]. This algorithm is capable of identifying unoccluded, partially occluded, and overlapping objects. The objects to be identified can have any location or orientation in the image plane, but it is assumed that their size (i.e. scale) is invariant. For more details on this technique, see [9].

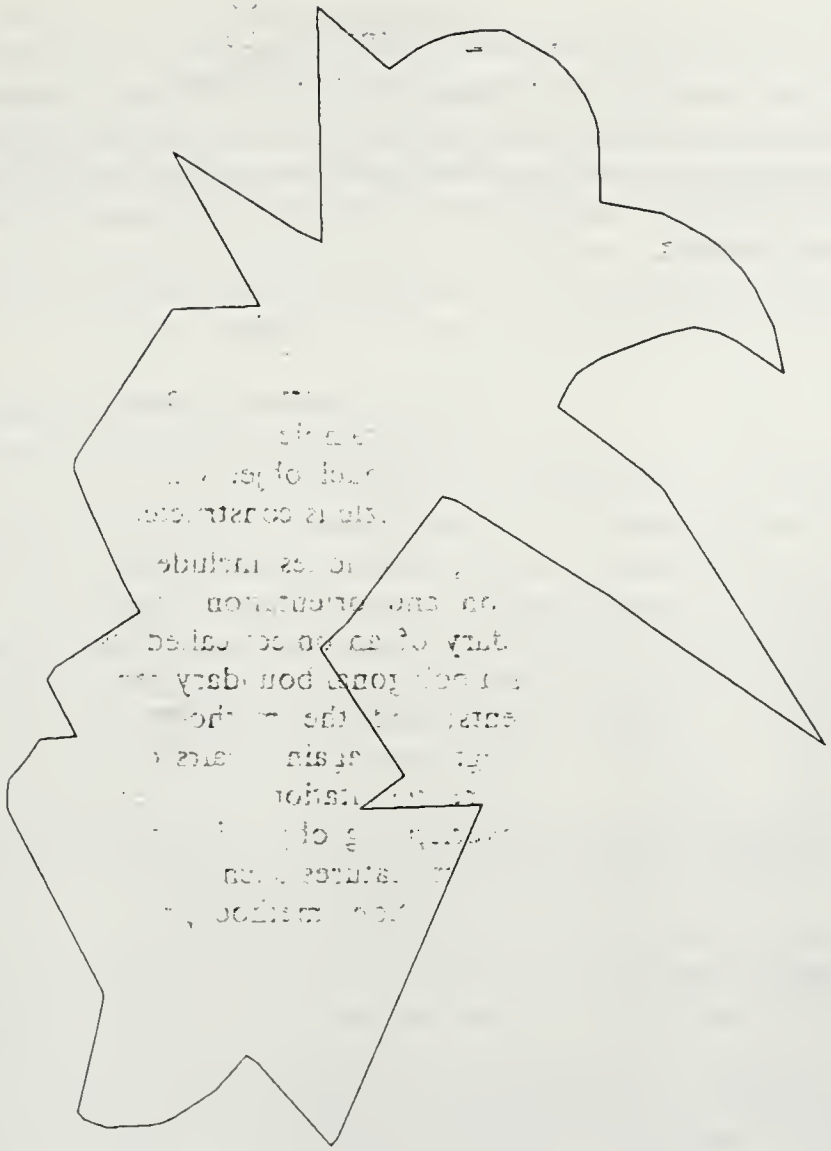


Figure 1.1. *A Puzzle Formed by Overlapping 2-Dimensional Objects.*

2.1. Identifying Unoccluded and Partially Occluded Objects

Once it has been acquired, the boundary curve of an object O is matched against a candidate model M as follows: Let the boundary curve of O be represented by a sequence $(U_j)_{j=1}^{n}$ of evenly spaced points lying on the curve, and let $(V_j)_{j=1}^{n}$ be the corresponding sequence of points representing M. Since the object might match to a subsequence starting anywhere along the model, we need to find the Euclidean transformation E of $(U_j)_{j=1}^{n}$ and a starting point k that minimizes the L_2 distance between the two sets of points in the least-squares sense; that is we need to find

$$\min_{k} \min_{E} \sum_{j=1}^{n} |EU_j - V_{j+k}|^2,$$

where the sum $j+k$ is calculated modulo the total number of points m representing M. The model for which this least-squares distance is minimized is then considered to be the best match for the observed object - i.e. the object is considered to be *identified* as being an instance of this best matching model.

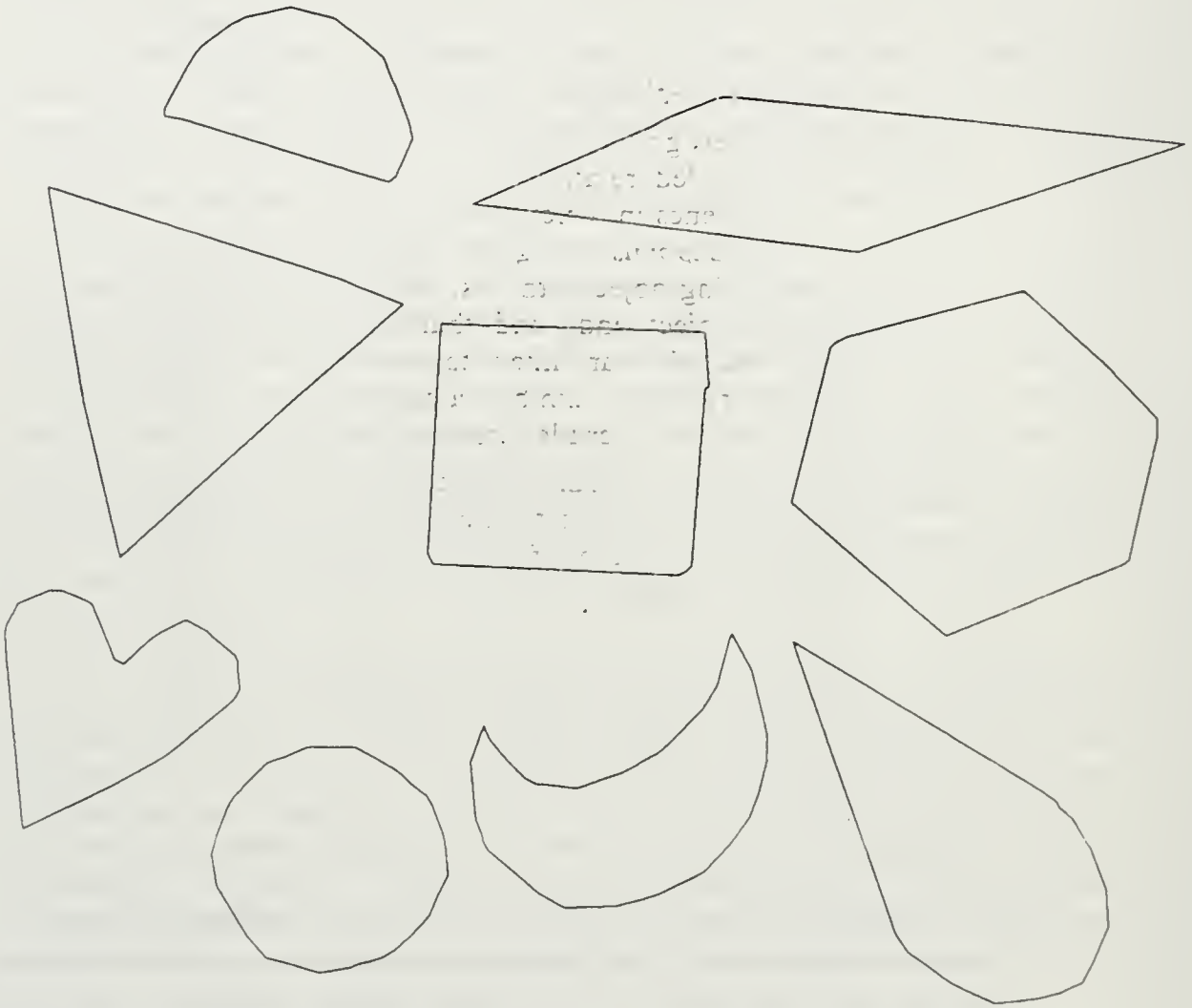


Figure 1.2. *9-Object Database Used To Form Puzzle.*

In this least-squares matching the Euclidean transformation E can be calculated explicitly, and can use the Fast Fourier Transform (see [9]). The cost of matching is therefore only $O(km \log m)$, where k is the number of candidate model objects, and m is the number of evenly-spaced points used to represent the boundary curve of each model.

For a match to be attempted, we require that $m \geq n$ (otherwise the observable portion of the boundary of O is too long to match M). This crude test eliminates a few candidate models without matching.

The matching algorithm just described often recognizes objects correctly even after occlusion of up to 85%-90% of their boundaries, even in the presence of *imposters* that are indistinguishable by human observers from actual objects in the model database. Of course, so high a degree of discrimination depends on the general nature of an object's boundary curve and on the set of models against which the object is being matched.

2.2. Recognizing Overlapping Objects

The method for recognizing a partially occluded object that we have just described can be extended relatively easily to allow for the recognition of objects in composite scenes in which a number of objects overlap. To do this we first find the *breakpoints* along the external boundary of a region R formed from overlapping objects, that is, those points at which the boundary of one overlapping object ends and that of another object begins. For obvious reasons, breakpoints are likely to be points at which the direction of the boundary of the region R exhibits a sharp concavity. Such concavities are found using the following crude technique:

(1) For some fixed value k, take k successive boundary points starting at each successive place j, and estimate the boundary tangent by calculating the line of best (least squares) fit to these successive points.

(2) Calculate the second derivative by differencing successive tangents, and look for maxima of the second derivative. (By ignoring minima of the second derivative, we bypass convexities, and find concavities only.)

Define the breakpoints to be these maxima. Once breakpoints have been found, the separate objects which have overlapped to form the region R are identified by matching each boundary section delimited by a pair of breakpoints against all candidate models, in the manner already described.

Two points concerning this breakpoint heuristic are worth noting. Our method will clearly fail if two bodies overlap without forming any points of concavity at the intersection of their boundaries. For example, two overlapping squares of equal size can accidentally take on the appearance of a rectangle. A person might also be fooled in a situation like this - though of course a person could sometimes use other information to infer that the rectangle is in fact two overlapping squares.

A second point is that it is always possible for false breakpoints to be inserted by the algorithm that we have described, since a single object may have a boundary convexity that cannot be distinguished from a breakpoint. This is a much less serious problem, since matching will still work correctly, whereas failure to recognize a real breakpoint will lead to a senseless attempt

to identify a composite formed from several objects with some one object in the database of models.

The breakpoints found for the puzzle shown in Figure 1.1 appear in Figure 2.1, and the analysis of this puzzle obtained using the least square matching algorithm is shown in Figure 2.2.

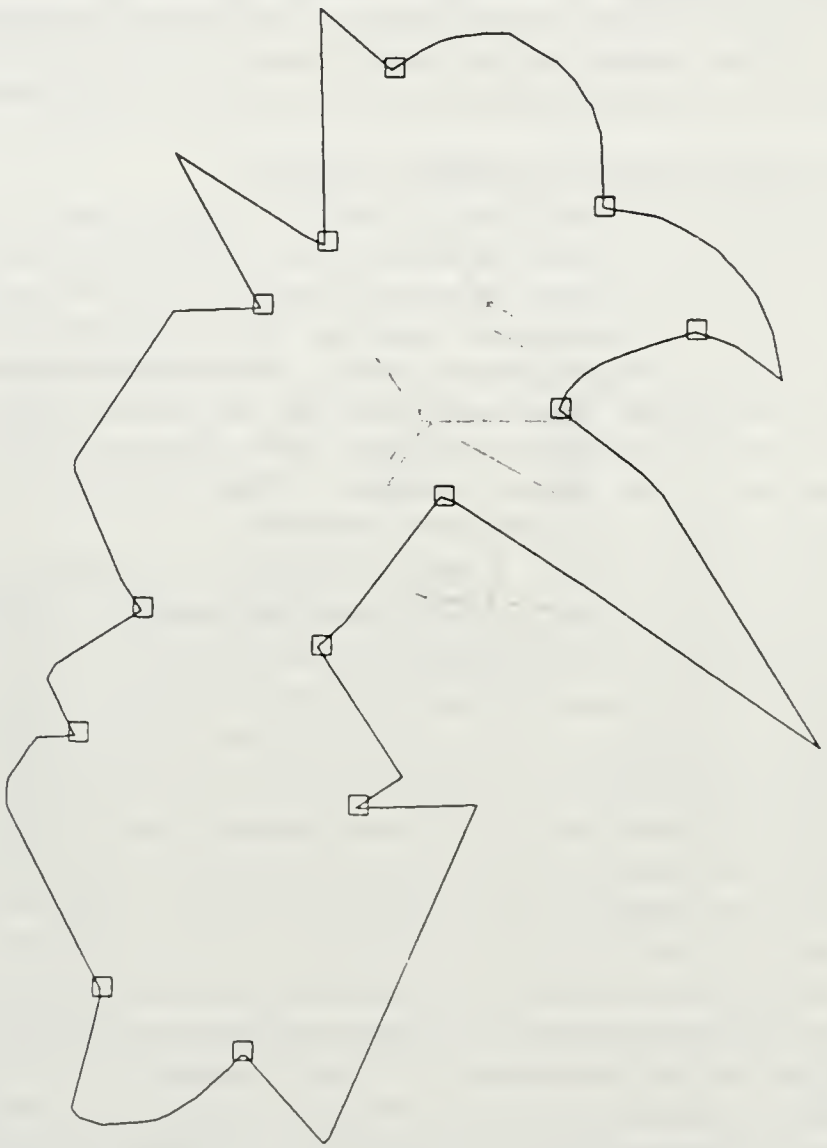


Figure 2.1. *Breakpoints Found in the Figure 1.1 Puzzle.*

Figure 2.2. *Solution Found for the Figure 1.1 Puzzle.*

Note that one false breakpoint found near the right-hand bottom portion of the puzzle results in the crescent shaped part of the boundary being matched twice. This illustrates a shortcoming of our present approach which is purely local, namely that no use is made of the global information that people would use to notice that adjacent (or non-adjacent) boundary sections may belong to the same object. For similar reasons, the square, parallelogram, and hexagon present in Figure 2.2 are matched twice. Note that the shorter unoccluded section of the hexagon, which is merely a straight line, is matched rather arbitrarily to the parallelogram. Use of an additional heuristic to eliminate models that extend beyond the boundary of a puzzle would eliminate the parallelogram in this case.

2.3. Preliminary Processing of Image Input

This section summarizes the pre-processing applied to all objects to obtain the evenly spaced boundary point representation needed by the matching algorithm described above. This involves the following steps:

(1) *Digitize Pictures.* A picture of one or more objects is digitized and stored as a 512x512 pixel image.

(2) *Find Connected Components.* A fast connected components algorithm is used to distinguish non-overlapping objects in the image. A box enclosing each object is found.

(3) *Record Boundary Information.* We locate the boundary of each object within its box, and store a polygon representation of this boundary. The center of gravity and area of each object is also computed; these latter features are not needed for matching *per se*, but serve for centering the object on the screen and for calibration.

(4) *Smooth Curves.* Since real world images are usually noisy, the arclength of bounding curves must be stabilized by using a smoothing process to eliminate boundary noise. The output of the smoothing operation is a simplified polygonal representation in which the number of object sides remaining is typically something like 10% of the number of boundary sides originally present.

(5) *Generate and Hash Footprints.* The footprint of each model object is obtained and a hash table is constructed. The smoothed polygonal representation is used as input.

(6) *Sample The Boundary.* The boundary of each object is discretized into the sequence of evenly spaced points needed for matching.

The smoothing step (4) is described in more detail in Section 2.4 below. The footprint step (5) is the topic of the subsequent sections.

2.4. Smoothing Noisy Curves

Boundary noise is eliminated using a fast (linear-time) algorithm to smooth an approximating polygonal curve. To see how this smoothing method works, first consider a curve that is monotonic in at least one coordinate direction. To smooth this curve, we place a narrow band around it, and find the shortest path between the curve's endpoints that lies completely within the band. (Informally, this shortest path can be viewed as the result of taking the curve as a piece of string and pulling it taut). The band around an observed curve within which we find the shortest path representing its smoothed version is just the ϵ-neighborhood of the curve, i.e. the set of points with distance at most ϵ from some point on the curve. The parameter ϵ is chosen in a manner reflecting our *a priori* knowledge of the amount of noise to be expected from our digitizing camera system.

This procedure can easily be extended to curves that are not monotonic in any one coordinate direction simply by decomposing such a curve into disjoint sections of monotonicity and applying the preceding method to each section separately. We actually use four directions for this purpose - the X-axis, the Y-axis, and the two lines at 45 degrees to these axes.

In eliminating boundary noise, the smoothing process introduces a slight source of additional error, as it tends to shrink objects and to smooth out subtle boundary features.

3. Selecting Candidate Models Using Footprints

A *footprint* of an object O is any curve F derived from O in a rotationally and translationally invariant manner, and additionally in a manner that only depends on some relatively local portion (*window*) of the boundary of O around each given starting point *s* on O's boundary. Such a curve F may be thought of as a crude geometric characterization of the boundary of the object. The space in which such a footprint exists may be of any dimension; the number of dimensions of this space is simply the number of parameters of shape formed from each local section of O. (The particular footprint that we have found adequate is 5-dimensional.) Footprints of model objects are stored in a hashed manner explained below. This allows the footprint of a partially obscured object to be hashed to retrieve the small set of models whose footprints closely match that of the puzzle; such hashing is of course fast and has a cost that is relatively independent of the number of objects in the model database. This enables us to use a large database of model objects without substantial performance degradation.

3.1. Footprint Generation

Footprints (or *signature* curves, [6]) can be generated in many ways. The mapping O → F of object to footprint need not be invertible. However, to serve our purposes, it should satisfy the following criteria:

(1) As mentioned above, the footprint must be invariant under rotation and translation since the object to be identified will generally not be presented in the same orientation and position as the model object.

(2) The footprint of a partially occluded object must be a portion of the footprint of the corresponding unoccluded object; i.e., the footprint must be generated from *local* boundary information only. For example, since such global data as area or center of gravity changes drastically when an object is partially occluded, such information is useless for identification of partially occluded objects.

(3) While not essential, it is desirable that objects with locally similar footprints should have local boundary features that the human eye takes as similar. This would make our artificial footprints correspond, in some ill-defined manner, to human intuition.

After experimenting with several different footprint constructions, we were most satisfied with the footprint recipe described just below. Object discrimination with this technique proved to be significantly sharper than with others tried and identification quite accurate.

The footprint in question is obtained from the *arclength vs. turning angle graph* G of an object (see [10] for a similar representation). This graph consists of a sequence of connected straight line segments, and is constructed as follows (arclength is plotted along the X-axis and turning angle along the Y-axis). Let $S_1,S_2,\ldots,S_n$ be the sides of a given object. For each side S_i we add to the graph G a horizontal line segment parallel to the X-axis of length proportional to the arclength of S_i, which is immediately followed by another straight graph segment whose length is proportional to the angle between side S_i and side S_{i+1}, i.e. the *turning angle* encountered at the vertex formed by S_i and S_{i+1}. This segment has a slope of 1 (resp. -1) if the turning angle is positive (resp. negative). (Since the arclength is measured in pixels while angles are measured in radians, segment lengths must be scaled appropriately to give sufficient weight to turning angles.) A sample graph G is shown in Figure 3.1.

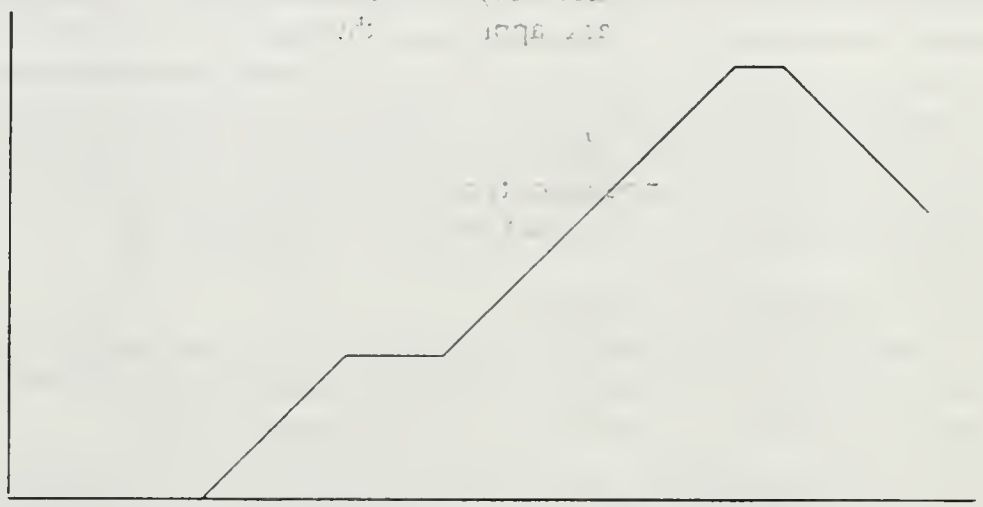


Figure 3.1. *Modified Arclength vs. Turning-angle Graph.*

To obtain a footprint from G, we divide the X-axis used to construct G into evenly spaced intervals, discretize the graph into a set $(U_j)_{j=1}^{n}$ of points, and form the first four sine and cosine coefficients from each section of G extending from a starting abcissa x to $x+ws$ (ws is our *window size*). This somewhat arbitrary construction maps each point U_j of the graph G to a corresponding footprint point. As a fifth coordinate for the footprint we use the total turning angle through which the original curve turns to the window section beginning at point U_j; the purpose of this fifth coordinate is to emphasize sharp features. Note that if the portion of the graph beginning at a point U_j is too short to allow a window of length ws, no footprint point

corresponding to U_j is generated. While this is unavoidable when the original object is not closed (e.g. a partially occluded object), it is important not to lose this information for model objects, which are always closed curves. Accordingly, the boundaries of model objects, and thus their corresponding arclength vs. turning angle graphs, are doubled so as to turn through 720 degrees.

Footprint curves generated in this way have the properties (1) and (2) described above. To see this, first observe that if we always generate the arclength vs. turning angle graph starting from the same point on the boundary of an object, the graph G which results will be independent of the orientation or position of the object. Moreover, if G' is the arclength vs. turning-angle graph of the same object formed starting from a different point on the boundary of O, then G' will simply be the graph G translated towards the origin, with an initial segment of G cut off and pasted to the end of G'. Hence if the boundary of O is doubled and the graphs G and G' extended in a corresponding fashion, every subgraph of the graph G' filling a window of size *ws* will match a translated section of G. Invariance then follows from the fact that the Fourier coefficients of a graph section (except for the 0-coefficient which we do not use) are unaffected if the graph is raised or lowered. The same remark applies to the total angle through which our original curve turns.

3.2. Footprint Hashing

Once generated, model footprints are hashed as follows. We divide 5-space into hypercubes of a fixed size. Every hypercube through which some portion of a model's footprint passes is stored in a hash table. With each stored hypercube we associate a list of all models whose footprint passes through the hypercube. Then, given the footprint F of an observed object (or portion of an observed object) to be identified, we retrieve only those models associated repeatedly with hypercubes that contain sections of F. These models are then *candidate* models.

If there were no noise in the curves or other sources of error, we could select as candidates only those models whose footprints pass through *every* hypercube that contains a section of F. However, unavoidable noise forces us to consider all models whose footprint passes through *relatively many* hypercubes that contain a section of F. The simple heuristic used to distinguish *many* from *few* is described in Section 4.1.

4. Experiments With Footprint-Based Matching

Experiments have been performed using model databases of sizes 9, 48, and 100. In our initial work with the 9 model database example shown in Figure 1.1, and found that a considerable amount of tuning was required to achieve good results from the footprints. However, subsequent experiments

with the larger databases yielded equally good results, and no further tuning was required.

4.1. Initial Footprint Refinement

During the initial footprint tuning process, various refinements to the basic footprint heuristic were introduced. Taken together, these refinements enhanced footprint effectiveness substantially.

Assigning A Match Rate. In order to eliminate candidate models whose footprints lie near the footprint of the puzzle at only a few points, we assign a *match rate* to each model, which serves as a score. More specifically, when attempting to identify a puzzle section O, a model M accumulates a *match point* whenever the footprint of O passes through a cube containing the footprint of M. The match rate for M is then:

$$\frac{NMP}{NFP}$$

where

NMP = Number of Match Points for M.
NFP = Number of Footprints Generated for O.

Note that the match rate can never be greater than 1, and the ideal situation is for the match rate for the correct model to be 1.

Match rate figures measure the robustness of the footprint technique, which is performing well if:

(1) The correct model has a high match rate.

(2) The match rate of all other models is 0 or at any rate significantly lower than the match rate of the correct model (except when the eye judges that there is substantial geometric similarity between some substantial feature of the incorrect model and a visible feature of the puzzle, in which case the match rate must be expected to be close to that of the correct model.)

Any footprint scheme that meets these criteria reasonably well can be used to eliminate models with low match rates. We have found it appropriate to eliminate all candidates whose match rate is lower than 25% of the highest candidate match rate. Since the candidate with the best match rate is generally that ultimately selected, another possibility is to process candidates in order of decreasing match rate, and to use the least square fit matching algorithm to determine when to stop - i.e., to stop when the model examined matches the puzzle sufficiently well in the least squares sense. Then, when the correct candidate has the highest match rate, which is usually the case, *and* the identification criterion works well, a match will be found immediately. A natural extension of this heuristic is to use the closeness of a least square match to determine when the footprint algorithm has failed to

find any reasonable candidates, in which case other analytic procedures should be tried.

Scaling of Parameters. The range of values over which each parameter entering into the footprint varies is characteristic of that parameter. To ensure that each such parameter is assigned equal evidential weight, we scale values in each dimension of footprint parameter space so that they all span identical ranges. The cube side length is then chosen as some fraction of this standard range.

Using Multiple Window Sizes. Choice of the window size ws used for the Fourier Transform involves a trade-off. If the window size is too small, significant global features of the boundary of an object may be missed. On the other hand, if it is too large, it may be impossible to form enough footprint points for a puzzle section for candidates to be selected with sufficient certainty. Moreover, no footprints at all will be generated for particularly short or featureless puzzle sections. To see interesting features of long puzzle sections without losing all ability to handle puzzle sections that are short or less interesting, we generate several sets of footprints using different window sizes. For each model object, all these sets are stored in the database. Then, given a puzzle object, we first try to perform a match using footprints generated with the largest window size. If no reasonable candidate is found, we try the next largest window size, and so on. If there is no footprint for the smallest window size, the puzzle segment is probably too small to try to match anyway.

Cube Centering. If a footprint section of a puzzle object O lies close to the boundary of a cube, it is possible that the corresponding footprint section of the correct model M is near the footprint of O, but actually lies in a neighboring cube. Thus we may miss matches because of discretization error. To better center puzzle object footprints, we proceed as follows. Model object footprints are hashed using a cube grid D where the length of a cube side is s. When processing a puzzle footprint, we shift the grid D by $\frac{1}{2}s$, obtaining a new grid $\hat{D}$. Each cube in $\hat{D}$ intersects 32 cubes in D. For each puzzle footprint point, we find the cube in $\hat{D}$ that contains it, and examine each of the 32 intersecting cubes in D to find candidate models. In effect this searches in cubes with side length equal to $2s$. However, this is not equivalent to using a fixed grid with cube side length equal to $2s$, since we have more freedom in choosing the coordinates of these larger cubes and can achieve appropriate centering.

Other plausible refinements have not yet been tried. For example, it may be desirable to give more emphasis to interesting features by weighting each match point by a factor $\frac{al}{cl}$, where al is the arclength of the section S of the object boundary that contributed to the corresponding footprint point, and cl is the chord length of S. Note also that the match rate as formulated

uses only positive evidence. Even if a model matches the puzzle for a section, but differs radically over another section, the negative evidence is not used. Negative evidence could be incorporated into the match rate by, for example, looking for the *absence* of a footprint for a model M in larger cubes, and subtracting points from M if the footprint is not close at all.

4.2. Parameter Adjustment

The greatest improvements in our original work with a small database were obtained by adjusting the parameters entering into the various stages of the matching process. The values chosen for parameters pertaining specifically to the effectiveness of the footprint algorithm are:

(1) *Arclength Scale Factor.* Objects typically have a perimeter of about 400 pixels, whereas the range of turning angles is $-\pi$ to π. When constructing the arclength vs. turning-angle curve, the arclength values are multiplied by a scaling factor to weight the angles appropriately. The scaling factor currently used is .03, e.g. a segment 33 pixels in length is assigned roughly the same evidential weight as an angle of 1 radian.

(2) *Step Size.* The step size used to discretize the arclength vs. turning-angle graph is .03, e.g. approximately 3 pixels or 5 degrees of angular turning. If this step size is too large, the footprint is coarsened; if too small, we generate more hash table entries and thus require additional storage space.

(3) *Window Size.* We experimented with window sizes of 70, 60, and 40 steps. (The window size is of course scaled by the step size.) With the arclength scale factor and step size given the values noted above, a window size of 60 generally discriminates well between object features of significantly different shapes as judged by the eye. Increasing the window size to 70 reduces the number of candidates selected for matching, but sometimes fails to include the correct model in the list of candidates. In analyzing overlapping puzzles it was therefore safer to start with a window size of 60. Footprints generated using window sizes of less than 60 discriminate significantly less well. Thus additional footprints generated using a window size of 40 serve primarily as a safety net, which can be used in those cases when there was no match at 60, to avoid searching the entire database. (If no hits are detected with this smaller window size, matching of the current boundary section is simply abandoned.) Normally such cases arise only for short, featureless boundary segments.

(4) *Cube Size.* The length of each side of the cubes into which our footprint parameter space is arbitrarily divided is approximately 1/32 of the total range of footprint values. This divides the total parameter space into roughly 32×10^6 cubes, of which all but roughly .1% are empty (on our hundred-model runs). As described above, the cube side length used

during the matching process is twice this, i.e. 1/16 the range of parameter values.

Other adjustable parameters of our procedure, not pertaining directly to footprints, include the width of the band used for smoothing (2 pixels); the sample step size used to discretize object boundaries for matching (6 pixels); and the concavity threshold value used to find breakpoints (5 degrees).

Other parameters have never been varied because good results are obtained without tuning them. For example, we never experimented with footprints formed using other than 5 parameters, even though sharper results might thereby be obtainable; conversely, the use of 5 parameters may be overkill.

4.3. Experiments With Large Vocabularies of Models

The 100 model database used in our experiments consists an assortment of polygonal and curved objects with perimeters ranging from 264 to 647 pixels. Of these 100 objects, 12 have more than one breakpoint. While these figures are easily distinguishable by the eye, there are many local boundary ambiguities among them. The 48 model database is a subset of the 100 model database.

We ran two types of experiments with these databases. First we selected 40 objects in the database to be used as puzzles - that is, a 40 object subset of the models was matched against both the 48 and 100 model databases; we then solved 7 complex puzzles where each puzzle consisted of from 6 to 8 overlapping objects. The results presented below were obtained with footprint window sizes of 60 and 40, and using the 25% of highest candidate match rate heuristic described in Section 4.1. All digitized images of models and puzzles are made using the same camera height and position.

In the first unoccluded object recognition experiment, the footprint algorithm has of course no identification problem and assigns the highest match rate to the correct model except in one instance out of 80. (The correct model usually does not have a match rate of 1.0 because we use different digitized images of model and puzzle objects in which there are differing noise errors introduced.) This experiment tests the footprint's ability to discriminate between the correct model and all others, determines whether using the strategies described in Section 4.1 are able to select a small set of candidates, and determines how much the number of candidates selected increases in going from 48 models to 100 models.

To assess powers of discrimination, we compare the match rates for the correct candidate models to the match rates of the candidates with the next highest match rate. The average match rate for the correct candidate is .79 using the 48 model database and .82 using the 100 model database. For the next most likely candidates, the average match rate is .24 using the 48 model database and .26 using the 100 model database. The average number of

candidates actually selected for matching to each of the 40 objects using the 48 object database is 2.5, while the number of candidates selected for matching each of the same 40 objects using the 100 model database is 3.4. Thus, the number of selected candidates is low in both cases, and its increase going to the larger database seems to be sublinear. Of course, these results are affected by to the average degree of similarity in the models used.

Results from overlapping puzzle experiments using the 100 model database are as follows. Whenever the visible boundary segment of a partially occluded object contains an interesting or significant feature of the object, identification is invariably and reliably made. However, erroneous identification sometimes results if the boundary segment is relatively featureless. In such cases the object matched generally has a feature quite similar to that of the visible portion in the puzzle section, e.g. of 49 puzzle segments tried, 34 were identified correctly. Of the incorrect or unattempted identifications, 1 resulted from a missing breakpoint, 2 resulted from too many breakpoints so that the remaining sections of the boundary were too short to identify, 2 resulted from an error made by the least square matching algorithm, and 9 resulted from the footprint algorithm failing to choose the correct model as a candidate. However in 9 of these last 11 cases, the object that was chosen is quite close to the correct solution. For the 34 correct identifications, the average number of candidates surveyed was 5.2. (The average number of candidates would be 1.1 if we use the least square matching algorithm to determine when to stop selecting candidates.)

As an example of what these figures mean, consider the puzzle in Figure 4.1. Figure 4.2 gives the correct solution. (Note that evidence from internal boundary segments are not included in the analysis.) Figure 4.3 gives the solution found using the 48 model database, and 4.4 gives the solution found using the 100 model database.

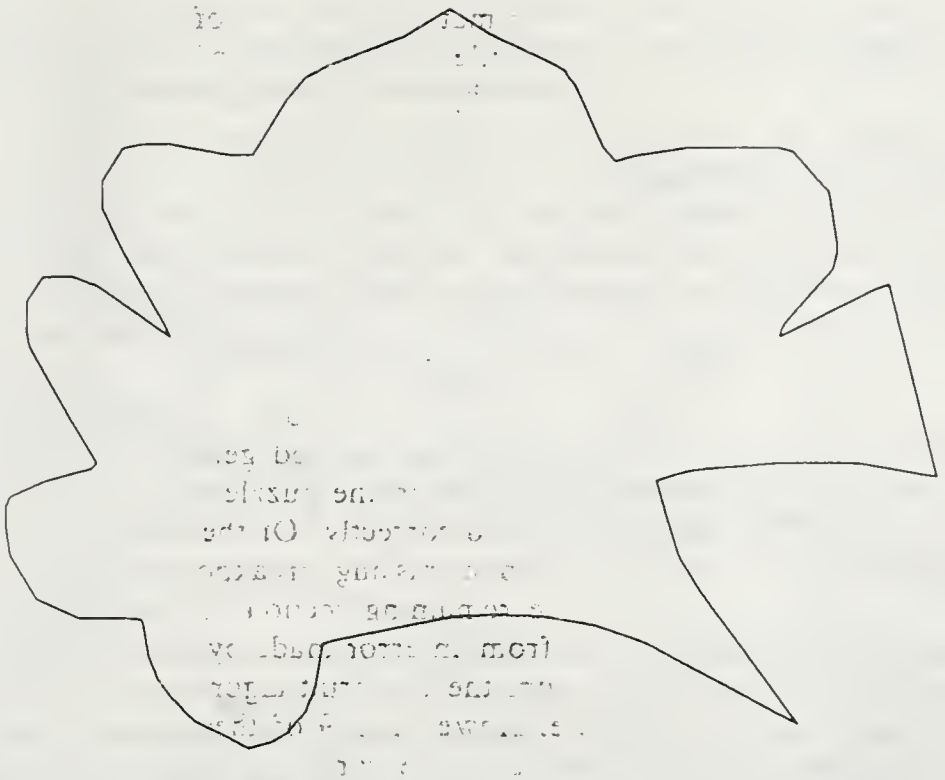


Figure 4.1. *Puzzle Run Against 48 and 100 Model Databases.*

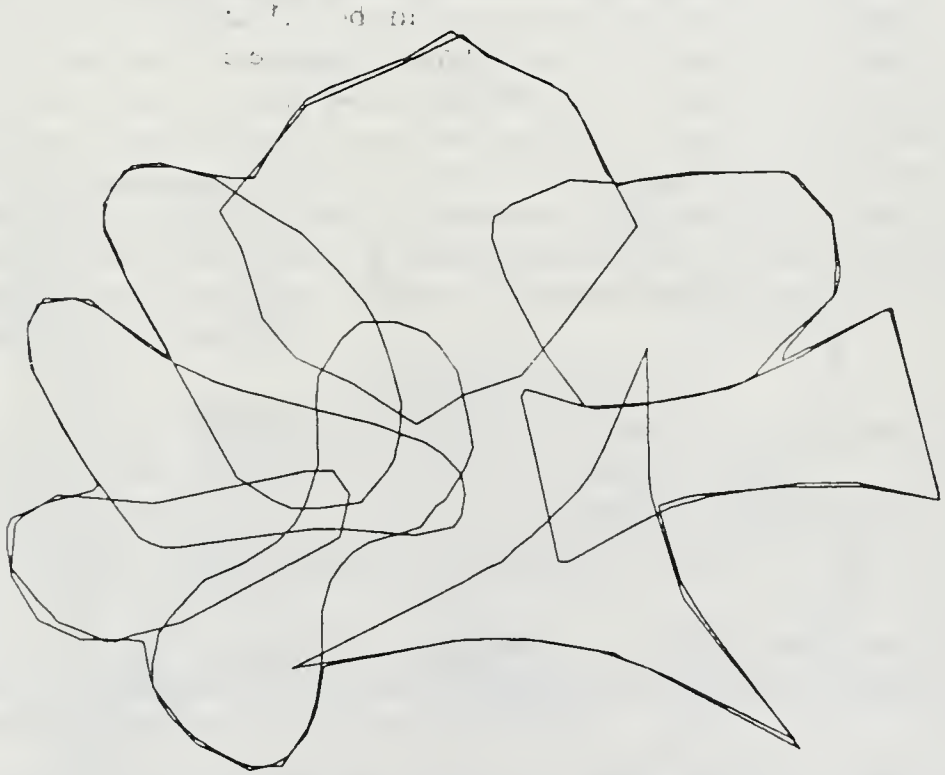


Figure 4.2. *Internal Construction of Puzzle.*

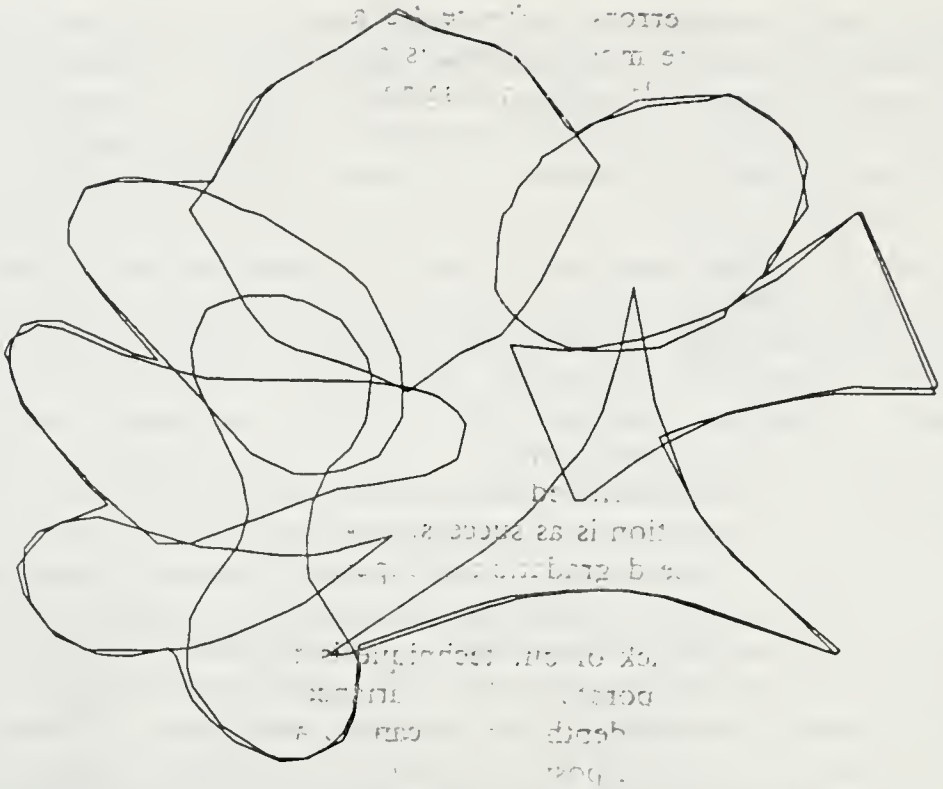


Figure 4.3. *Solution Found by Matching Against 48 Model Database.*

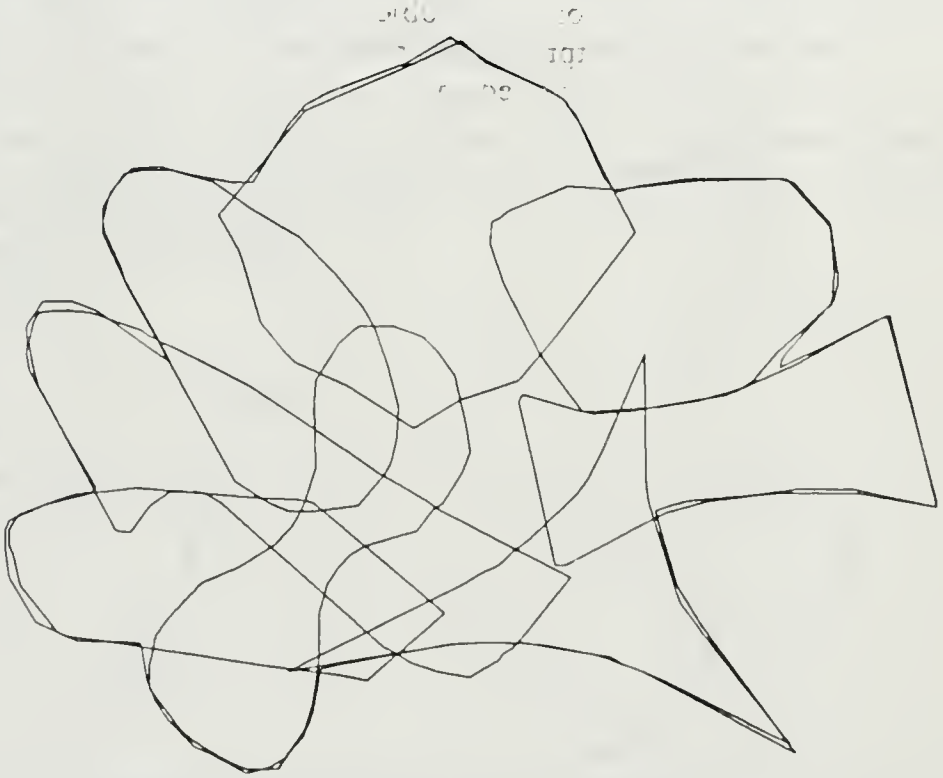


Figure 4.4. *Solution Found by Matching Against 100 Model Database.*

Note that the two errors in Figure 4.3 and in Figure 4.4 are cases in which the boundary of the model chosen is so close to the visible puzzle segment that a person might also make the same mistake.

It is also worth noting that performance degrades only slightly going from the smaller to the larger database. Analysis of the puzzle shown in Figure 4.1 requires 90.9 seconds of execution time using the smaller database, and 100.3 seconds using the 100 model database. The number of cubes (representing 2 window sizes) stored in these two cases are 18609 and 40967 respectively.

5. Conclusions

The use of footprint hashing as a method for quickly selecting candidates for matching to unoccluded and overlapping objects seems to be highly effective. Identification is as successful with 100 models as with 48, and only a slight performance degradation is experienced when using this significantly larger database.

A major drawback of our technique is that it is not scale invariant. We choose not to incorporate scale invariance in the footprint and matching algorithms because a depth sensor can be added to a vision system to give the (X,Y,Z)-coordinate positions of an object, without any ambiguity of size entering. Another problem is that the breakpoint heuristic fails when either models have concavities or when objects overlap without forming a sharp point of concavity. Footprint evidence may be particularly useful in this regard, since it can be employed to distinguish breaks in the boundary using information about objects in the database rather than a purely local object independent criterion. Finally, we remark that our techniques extend easily to 3-dimensional objects on which rotationally invariant curves can be found. Of course, global heuristics, e.g. recognition of cases in which several non-adjacent sections of a puzzle are parts of the same object, need to be added to the techniques we employ.

REFERENCES

[1] N. Ayache, J. Boissonat, B. Bollack, B. Faverjon, "Automatic Handling of Overlapping Workpieces", *I.C.P.R.*, Montreal, 1984.

[2] S. Berman, P. Parikh, C.S.G. Lee, "Computer Recognition of Two Overlapping Parts Using a Single Camera", *Computer*, Vol. 18, No. 3, Mar. 1985, pp. 70-80.

[3] D. Bhanu, O. Faugeras, "Shape Matching of Two-Dimensional Objects", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-6, No. 2, Mar. 1984, pp. 137-155.

[4] R. Bolles, "Robust Feature Matching through Maximal Cliques", *Proc. SPIE Tech. Symp. Image and Assembly*, Washington, D.C., Apr. 1979.

[5] R. Bolles, R. Cain, "Recognizing and Locating Partially Visible Objects: The Focus Feature Method", *The International Journal Of Robotics Research*, Vol. 1, No. 3, 1982, pp. 57-81.

[6] P. Coiffet, "Interaction With the Environment", *Robot Technology*, Vol. 2, pp. 123-181, Kogan Page Ltd., London, England, 1983.

[7] M. Koch, R. Kashyap, "A Vision System to Identify Occluded Industrial Parts", *IEEE International Conf. on Robotics and Automation*, Mar. 1985, pp. 55-60.

[8] W. Perkins, "A Model-Based Vision System for Industrial Parts", *IEEE Transactions on Computers*, Vol C-27, No. 2, Peb. 1978, pp. 126-143.

[9] J.T. Schwartz, M. Sharir, "Identification of Partially Obscured Objects in Two and Three Dimensions by Matching of Noisy *Characteristic Curves*", Technical Report, Robotics Activity, New York University, 1985.

[10] J. Turney, T. Mudge, R. Volz, "Recognizing Partially Hidden Objects", *IEEE International Conference on Robotics and Automation*, Mar. 1985, pp. 48-54.

This book may be kept

FOURTEEN DAYS

A fine will be charged for each day the book is kept overtime.

JUL 8 0 1991

GAYLORD 142 PRINTED IN U.S.A.

NYU COMPSCI TR-162
Kalvin, Alan
c.1

Two dimensional model
[illegible]ndary

NYU COMPSCI TR-162
Kalvin, Alan
c,1

Two dimensional model
based boundary

DATE DUE | BORROWER'S NAME

G Davis

LIBRARY
N.Y.U. Courant Institute of
Mathematical Sciences
251 Mercer St.
New York, N. Y. 10012